```
#include <stdio.h>
int exchange(int *a, int i, int j){
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
int partition(int *a, int p, int r){
    int x = a[r];
    int i = p-1, j;
    for(j = p; j < r; j++){
        if(a[j] < x){
            i++;
            exchange(a, i, j);
        }
    }
    exchange(a, i+1, r);

    return i+1;
}
int quick_sort(int *a, int p, int r, int k, int *find)
{
    if(p < r){
        int q = partition(a, p, r);
        printf("主元素q=%d\n", q);

        if(k == q){
            *find = 1;
            printf("find number k = %d\n", a[q]);
            return 0;
        }
        else if(k < q){
            quick_sort(a, p, q-1, k, find);
        }
        else if(k > q){
            quick_sort(a, q+1, r, k, find);
        }
    }
}

 int main()
 {
    int find = -1, k;
    scanf("%d", &k);
    int a[] = {0,9,7,5,3,6,1};      //试验数组

    quick_sort(a, 0, 6, k, &find);  //因为在整个递归过程中，存在q与k一直不相等的情况（实际上就是
                                    //递归到底时，q恰好就是是k的相邻元素）。
                                    //不过无妨，到了最后，即便整个数组可能不是有序的，但是q和k所
//对应的数，已经恰好是第q大和第k大的数了，目的达成
    if(find == -1){                 //这时只需将对应下标k的数找出即可
```

```
        printf("find number k = %d\n", a[k]);
     }

/*打印当前数组元素的顺序 */
    int i;
    for(i = 0; i <= 6; i++){
        printf("%d ", a[i]);
    }
 }
```